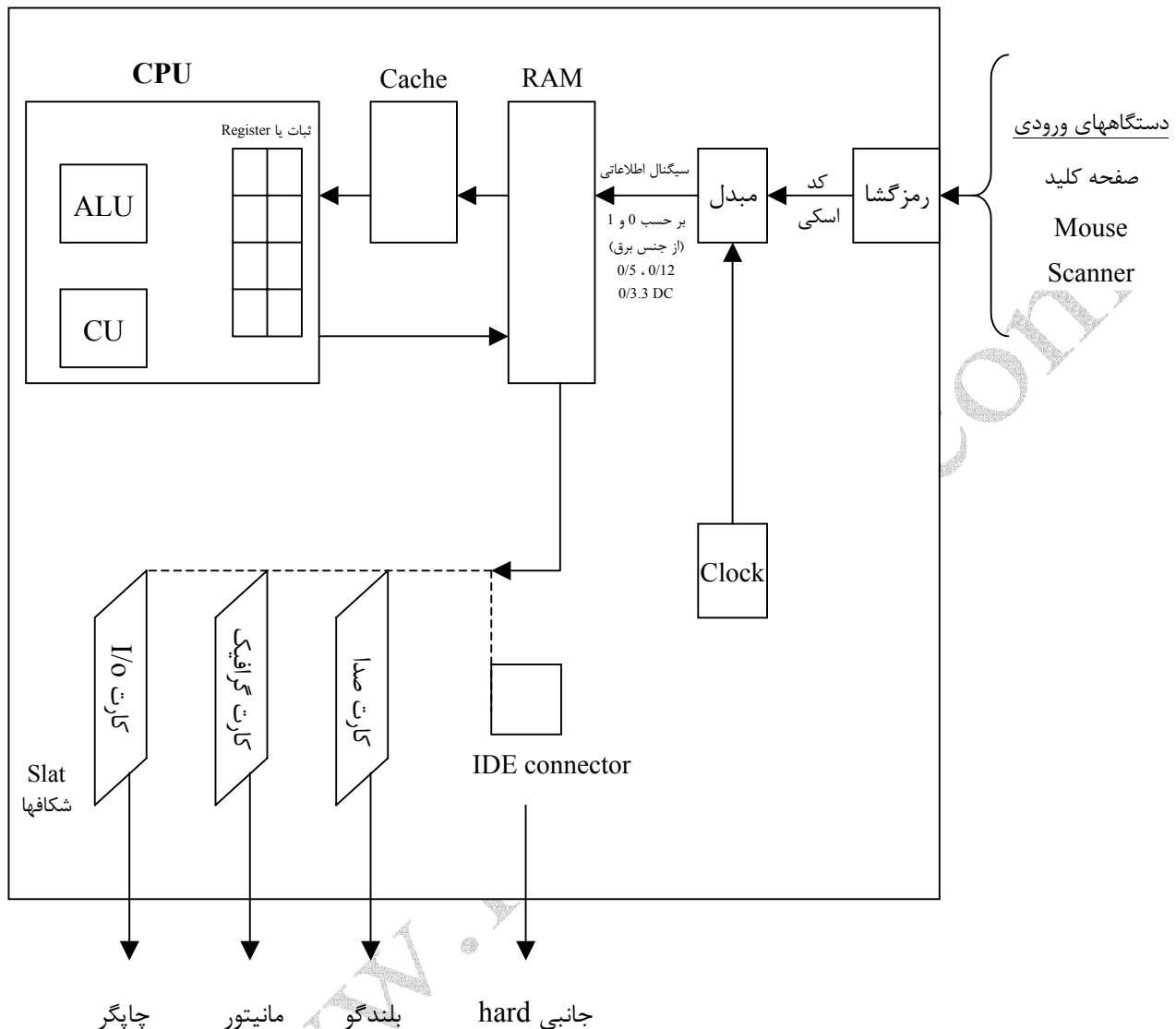


آشنایی با سیستم درونی و عملکرد داخلی کامپیوتر

Mother Board



تذکر ۱: کلیه اطلاعات ورودی از طریق بردهای کامپیوتر به درون کامپیوتر (مادربرد) منتقل می‌شوند. این اطلاعات ورودی درون کامپیوتر تبدیل به سیگنالهای اطلاعاتی می‌شوند که بر حسب صفر و یک است و از جنس ولتاژ و برق می‌باشد.

تذکر ۲: تولید سیگنالهای اطلاعاتی به کمک قطعه‌ای به نام **Clock** یا ساعت زمانبندی و تایمینگ انجام می‌شوند.

تذکر ۳: حافظه **RAM** یک حافظه الکترونیکی است که از تعدادی سلول برای نگهداری اطلاعات استفاده می‌کند. هر سلول یک مدار فلیپ فلاپ می‌باشد.

تذکر ۴: هر فلیپ فلاپ تنها می‌تواند مقدار صفر و یک را در خود نگهدارد.

- تذکر ۵: واحد نگهداری اطلاعات در هر سلول یا فلیپ فلاپ یک بیت است.
- تذکر ۶: اطلاعات ورودی از RAM به درون حافظه Cache می‌آید و سپس از طریق آن برای پردازش به CPU منتقل می‌شود.
- تذکر ۷: CPU پس از پردازش، اطلاعات را برای انتقال به دستگاههای خروجی، به حافظه RAM بازمی‌گرداند.
- تذکر ۸: حافظه RAM اطلاعات خروجی را به کارتهای واسط که درون Slatها قرار دارند می‌فرستد.

سؤال: Bus چیست؟

- جواب: به کلیه مسیرهای ارتباطی بین قطعات درون کامپیوتر Bus می‌گویند.
- تذکر ۱: در کامپیوتر ۳ نوع Bus وجود دارد:
- ۱- **Address Bus**: نشانی و آدرس محل‌های مختلف منتقل می‌شوند.
 - ۲- **Control Bus**: از طریق این مسیر، دستورات و فرامین منتقل می‌شوند.
 - ۳- **Data Bus**: کلیه داده‌ها و اطلاعات از طریق آن منتقل می‌شوند.
- تذکر ۲: مسیرهای Bus مسیرهای دوطرفه غیر همزمان می‌باشند.
- تذکر ۳: واحد سنجش قدرت Bus، بیت می‌باشد. بدین معنی که چه تعداد بیت هر Bus می‌تواند به طور همزمان منتقل نماید.
- تذکر ۴: واحد کنترل وظیفه دارد که بگوید کدام قطعه چه موقع چه کاری را انجام دهد.
- تذکر ۵: در سیستم بدن انسان، نخاع و مخچه نقش و رل CU را اجرا می‌کنند.
- تذکر ۶: ثبتانها، حافظه‌های موقت و سریع می‌باشند که در محاسبات از آنها استفاده می‌کنیم.

سؤال: Cache چیست؟

- جواب: حافظه‌ای است سریع و گران قیمت که در کامپیوتر کاربرد فراوانی جهت سرعت بخشیدن دارد. این حافظه باعث می‌گردد حالت انتظار در CPU پائین بیاید و CPU با سرعت بیشتری دستورات و فرامین را اجرا نماید.
- تذکر ۱: کلیه اطلاعات که در کامپیوتر منتقل می‌شوند بصورت سیگنالهایی هستند که از مبنای 2 تولید شده‌اند. وزن این اعداد در مبنای 2، NBCD می‌باشد.
- تذکر ۲: کلیه اطلاعات درون کامپیوتر در مبنای 16 ذخیره‌سازی می‌شوند.

تمرین : مقدار 65 - را در قالب 8 بیتی محاسبه نمائید.

$$01000001 \implies +65$$

$$10111110 \implies \text{مکمل یک}$$

$$\underline{\quad\quad\quad + 1}$$

$$10111111 \implies \text{مکمل دو} = -65$$

نکته ۲: در هنگام تبدیل داده‌ها و اطلاعات ، هر بیت دارای ارزشی می‌باشد که شما باید این حالت ارزش را در نظر بگیرید.

نکته ۳: در هنگام انجام محاسبات ، انتقال بیت از بیت سوم به بیت چهارم را Auxiliary می‌گویند در صورتی که انتقال بیت هشتم به بیت نهم را Cary می‌گویند.

		Ax	
Cary	Auxiliary	Cary = 0	11(1)
0010	0110+	0010	1001+
0001	0010	0001	1001
0011 1000		0100 0010	
<div style="display: flex; justify-content: center; align-items: center;"> <div style="border-left: 1px solid black; border-right: 1px solid black; width: 100px; height: 15px; margin-right: 5px;"></div> نیبل </div>			
یک بایت			

		Ax	
Cary	(1) 1(1) 11		
1001	0011+		
1001	1110		
10011 0001			

تذکر : مقدار Auxiliary و مقدار Cary پس از انجام محاسبات در ثبات Flag تنظیم و ست (set) می‌گردد.

Flag Register ثبات می‌باشد که به شما وضعیت پردازش را نشان می‌دهد.

نکته ۴: در برخی از محاسبات، حالت سرریز یا Over flow رخ می‌دهد که باعث می‌گردد ثبات Flag بیت OF آن به یک یا صفر تغییر یابد.

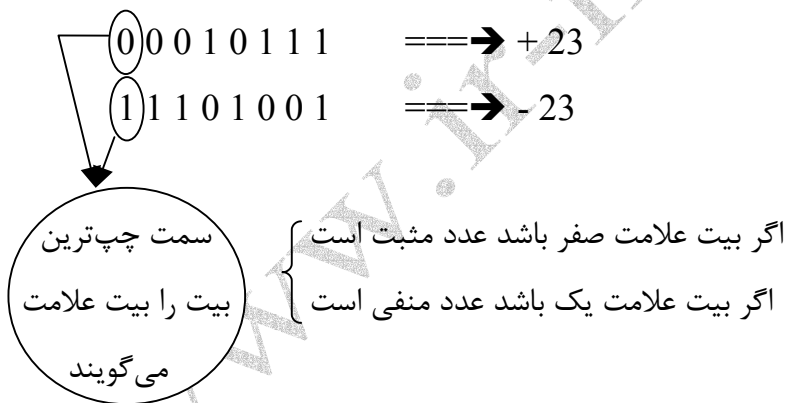
$$\begin{array}{r}
 \text{Cary} \quad \textcircled{1} \\
 10010011 + \\
 01010011 \\
 \hline
 11100110 \quad \left. \begin{array}{l} \text{OF} = 0 \\ \text{Cary} = 0 \end{array} \right\}
 \end{array}
 \quad
 \begin{array}{r}
 10010011 + \\
 10111010 \\
 \hline
 \textcircled{1}01001101 \quad \left. \begin{array}{l} \text{OF} = 1 \\ \text{Cary} = 1 \end{array} \right\}
 \end{array}$$

سرریز
(Over flow)

تذکر: حالت Over flow و Cary در برخی از موارد بطور همزمان رخ می‌دهد که شما می‌توانید برخی از حالت‌های Cary را در نظر نگیرید و تنها بصورت یک Warning (هشدار) باشد. همانند مثالهای فوق.

سؤال: بیت علامت چیست؟

جواب: به سمت چپ‌ترین بیت، بیت علامت می‌گویند که نشان دهنده نوع عدد است.



نکته ۵: با توجه به نوع محاسبه و داده‌های ورودی شما می‌توانید دستورات مورد نیاز را مشخص نمائید. بر خلاف خیلی از زبانها، در زبان ماشین فرمان یا دستور در ابتدای سطر قرار می‌گیرد که می‌تواند به اندازه یک یا دو بایت باشد که در اصطلاح به آن OP - Code می‌گویند.

نکته ۶: داده‌ها و محاسبات می‌توانند به صورت با علامت و یا بدون علامت باشند. اگر با علامت باشند سمت چپ‌ترین بیت به عنوان داده لحاظ نمی‌شود و چنانچه بدون علامت باشند، سمت چپ‌ترین بیت را بیت داده‌ای فرض کرده و به آن ارزش می‌دهند.

نکته ۷: با توجه به داده‌هایی که در نظر می‌گیرد از لحاظ با علامت بودن یا بدون علامت بودن رنج داده‌ای نیز تغییر می‌کند. همانند دو حالت زیر

128	64	32	16	8	4	2	1
0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1

داده با علامت 8 بیت + 127 - 127

داده بدون علامت 8 بیت 255 0

نکته ۸: با توجه به نوع دستورات ، داده‌ها تعیین می‌شوند و بر عکس با توجه به نوع داده‌ها ، دستورات مربوطه را باید استفاده نمائیم.

ADD ,

AAD ,

نکته ۹: برای شناسایی این نکته که آیا داده عددی زوج است یا فرد باید به بیت سمت راست داده توجه نمائید. اگر این بیت صفر باشد داده زوج است و چنانچه یک باشد این داده فرد است.

نکته ۱۰: در مبنای 10 دو نوع مکمل وجود دارد :

الف) مکمل 9: برای بدست آوردن مکمل 9 کافی است تمام ارقام را از عدد 9 کم کنید.

مکمل 9

$(46)_{10} \longrightarrow 53$

ب) مکمل 10: برای بدست آوردن مکمل 10 شما باید رقم اول را از عدد 10 کم کنید و الباقی ارقام را از عدد 9 کم کنید.

مکمل 10

$(46)_{10} \longrightarrow 54$

نکته ۱۱: در هنگام انجام محاسبات ، در نظر گرفتن مکمل 10 یا مکمل 9 کاملاً اختیاری است و شما می‌توانید بر حسب نیاز روش دلخواه خود را انتخاب نموده و به جواب درست برسید.

مثال :

$$\begin{array}{r}
 56 + \\
 \hline
 86 \xleftarrow{\text{مکمل 9}} - 13 \xrightarrow{\text{مکمل 10}} 87 \\
 \hline
 \textcircled{1}42 \qquad \qquad \qquad 43 \qquad \qquad \qquad \textcircled{1}43 \\
 \downarrow \text{بیرون} \\
 \hline
 1 + \\
 \hline
 43
 \end{array}$$

تمرین :

$$\begin{array}{r}
 93 + \\
 \hline
 57 \xleftarrow{\text{مکمل 9}} - 42 \xrightarrow{\text{مکمل 10}} 58 \\
 \hline
 \textcircled{1}50 \qquad \qquad \qquad 51 \qquad \qquad \qquad \textcircled{1}51 \\
 \downarrow \text{بیرون} \\
 \hline
 1 + \\
 \hline
 51
 \end{array}$$

تذکره : در دو مثال بالا مؤلفه دوم از مؤلفه‌های اول کوچکتر می‌باشد. اگر مؤلفه دوم بزرگتر از مؤلفه اول باشد باید طبق الگوی زیر عمل نمائید.

$$\begin{array}{r}
 73 + \\
 \hline
 14 \xleftarrow{\text{مکمل 9}} - 85 \xrightarrow{\text{مکمل 10}} 15 \\
 \hline
 87 \downarrow \qquad \qquad \qquad - 12 \qquad \qquad \qquad 88 \downarrow \\
 \hline
 \text{مکمل 9} \qquad \qquad \qquad \qquad \qquad \qquad \text{مکمل 10} \\
 \hline
 12 \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad 12
 \end{array}$$

نکته ۱۲: در برخی از محاسبات، رویداد Over flow و Cary دارای اهمیت می‌باشد و در برخی از محاسبات، دارای اهمیت نمی‌باشد. مثال زیر نشان می‌دهد که حالت Cary flag دارای ارزش نیست و شما می‌توانید آنرا در نظر نگیرید زیرا دو داده جزء داده‌های با علامت می‌باشد.

$$\begin{array}{r}
 65 \rightarrow 01000001 \\
 -65 \rightarrow 10111111 \\
 \hline
 0 \quad \textcircled{1}00000000 \\
 \text{CF} = 1 \qquad \text{OF} = 1
 \end{array}$$

با علامت

تمرین : محاسبات زیر را انجام دهید و بگوئید کدام یک از flagها ست می شود.

$$\begin{array}{r} 35 + \implies 00100011 \\ 24 \implies 00011000 \\ \hline 00111011 \end{array} \left. \vphantom{\begin{array}{r} 35 + \\ 24 \end{array}} \right\} \begin{array}{l} CF = 0 \\ OF = 0 \\ Ax = 0 \end{array}$$

$$\begin{array}{r} 97 + \implies 01100001 \\ 59 \implies 00111011 \\ \hline 10011100 \end{array} \left. \vphantom{\begin{array}{r} 97 + \\ 59 \end{array}} \right\} \begin{array}{l} CF = 0 \\ OF = 0 \\ Ax = 0 \end{array}$$

$$\begin{array}{r} 135 + \implies 10000111 \\ -24 \implies 11101000 \\ \hline 10110111 \end{array} \left. \vphantom{\begin{array}{r} 135 + \\ -24 \end{array}} \right\} \begin{array}{l} CF = 1 \\ OF = 1 \\ Ax = 0 \end{array}$$

سؤال : بیت توازن چیست؟

جواب : هرگاه بخواهیم در هنگام انتقال اطلاعات کنترل کنیم که داده‌ها و بیت‌ها درست منتقل شده‌اند یا نه از بیت توازن استفاده می‌کنیم. شما می‌توانید بیت توازن را همانند یک روش فرض نمائید. این بیت در بیت flag قرار دارد. در بیت توازن بدینصورت مطرح می‌کنیم که اگر تعداد یک‌های ارسالی فرد باشد بیت توازن یک است و چنانچه تعداد یک‌های ارسالی زوج باشد بیت توازن صفر است.

نکته ۱۳: در کامپیوترهای مدل 8086 و 8088 پهنای Bus (Data Bus) دیتا 16 بیت می‌باشد در صورتیکه پهنای Bus (Address Bus) آدرس 20 بیت می‌باشد.

$$m = 2^n \longleftrightarrow \text{Log}_2^m = n$$

تعداد خانه‌های حافظه (حالتها) تعداد بیت‌های مورد نیاز بر حسب صفر و یک

$$65536 = 2^{16}$$

اندازه } 64 KB
Segment

$$1024 \times 1024 = 2^{20} = 1 \text{ MB}$$

تذکر : ساختار Address Bus و Data Bus به صورت زیر می باشد :

	8088	8086
Data	8	16
Add	16	16
<hr/>		
Data	16 (به همراه قرض)	16
Add	20 (۴ بیت قرضی)	20 (۴ بیت قرضی)

- کامپیوترهای مدل **80286** : در کامپیوترهای 80286 می توان 16 میلیون بایت از حافظه را آدرس دهی نمود. بدین معنی که دارای 16 خط دیتا و 20 خط آدرس می باشد. بر خلاف کامپیوترهای مدل قبلی ، این کامپیوتر دارای 2 مُد آدرس دهی می باشد.

Real mode	مُد واقعی
Protect mode	مُد حفاظت شده

- کامپیوترهای مدل **80386** : در کامپیوترهای مدل 80386 اندازه ثابت ها ، 32 بیتی می باشد و همچنین پهنای Bus آنها 32 بیتی می باشد. در این مدل از CPU ها علاوه بر 2 مُد آدرس دهی قبلی ، حالت یا مُد مجازی نیز ارائه شده است.

Virtual mode	مُد مجازی
--------------	-----------

- کامپیوترهای مدل **80486** : این مدل از کامپیوترها دارای ثابت های 32 بیتی و پهنای Bus نیز 32 بیتی می باشد. در این کامپیوترها به اندازه 8 KB حافظه پنهان یا Cache نیز وجود دارد. این حافظه Cache بیرون CPU قرار گرفته است.

- کامپیوترهای پنتیوم : این مدل از CPU ها دارای ثابت های 32 بیتی می باشد که پهنای Bus آنها 64 بیتی است و تمام حافظه پنهان درون CPU ها قرار گرفته است.

تذکر : علت تنوع مدل پنتیوم در Pentium II و Pentium III مستقل بودن گذرگاهها می باشد و همچنین قطعه ای است که محاسبات اعداد اعشاری را انجام می دهد.

آشنایی بیشتر و دقیق‌تر با کدهای اسکی

کاراکتر	مبنای ۱۰	مبنای ۱۶		
A	→ 65	→ 41	NBCD →	0 1 0 0 0 0 1
a	→ 97	→ 61	→	0 1 1 0 0 0 1
کلید tab	→ 9	→ 9	→	0 0 0 0 1 0 0 1
0	→ 48	→ 30	→	0 0 1 1 0 0 0 0
2	→ 50	→ 32	→	0 0 1 1 0 0 1 0

این الگوی کد گذاری مبنای تولید سیگنال است

زبان ماشین

در هنگام انجام فعالیت به روی داده‌های عددی، رعایت کردن نکته‌ای که در مثال زیر بیان شده است مهم می‌باشد زیرا برای بدست آوردن نتیجه درست باید عدد 30 را از کد خارج نمائیم.

2 + 3
B8 02 03

Add 2, 3

3 2

حافظه 33 32

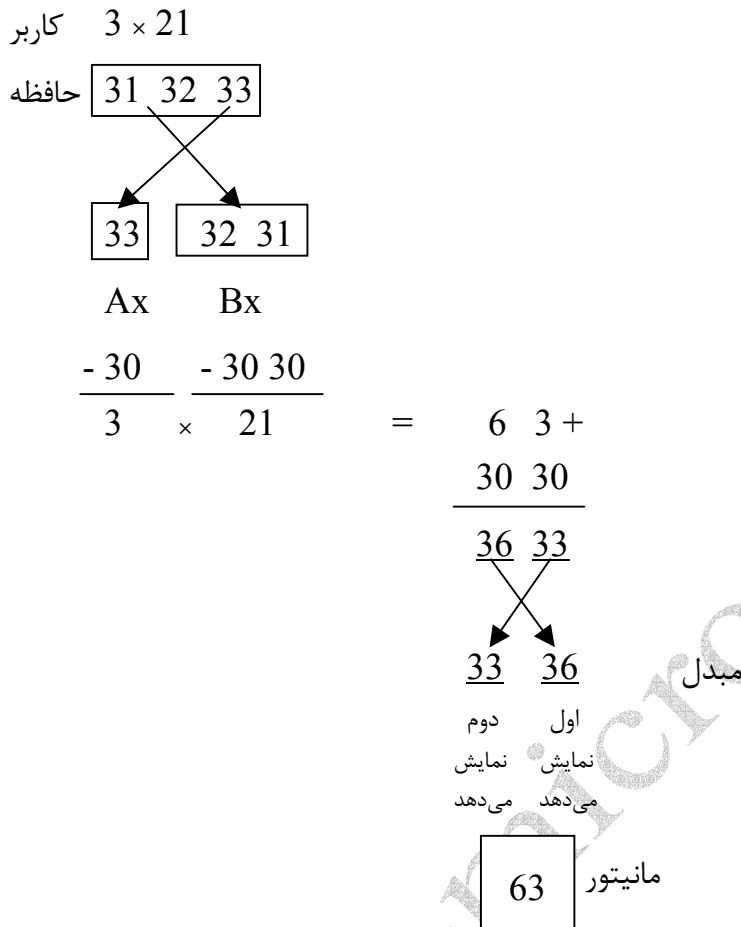
32 33

Ax Bx

-30 -30

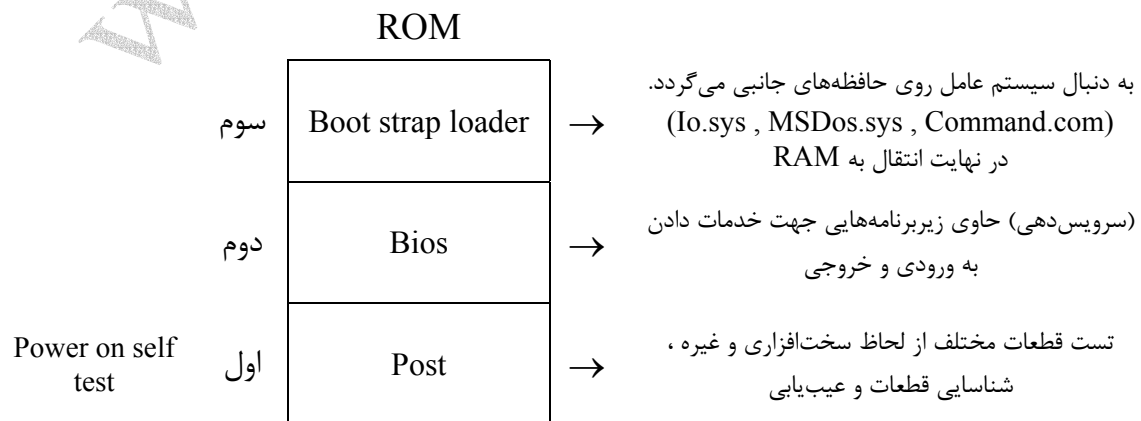
2 + 3 ==> 5 + 30 = 35 → نشاندهنده کاراکتر 5 می‌باشد

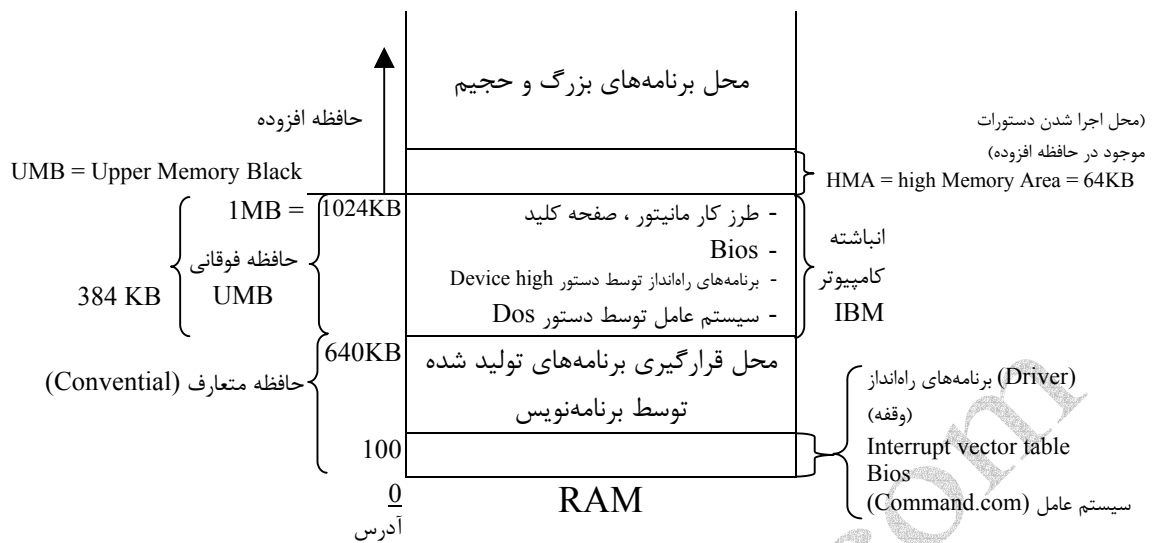
تمرین : فرآیند انجام محاسبه 3×21 را نشان دهید.



اطلاعات ورودی و خروجی کاراکتر است اما درون کامپیوتر بصورت کد اسکی در مبنای 16 می باشد که به شکل سیگنالهای اطلاعاتی در مبنای 2 و با وزن NBCD تولید می شوند.

آشنایی با ساختار حافظه RAM





تذکره ۱: سیستم عامل پس از بارگذاری در حافظه RAM ، دو فایل Config.sys و Autoexec.Bat را اجرا می‌کند و تمام دستوراتی که در این فایل وجود دارد را اجرا می‌کند. دو تا از مهمترین دستورات در فایل Config.sys عبارتند از :

himem.sys ✓

EMM386.EXE ✓

تذکره ۲: کلید برنامه‌هایی که در محیط IBM قرار است اجرا شوند باید در حافظه متعارف قرار گیرند و یا اینکه بصورت حافظه متعارف شبیه‌سازی گردند.

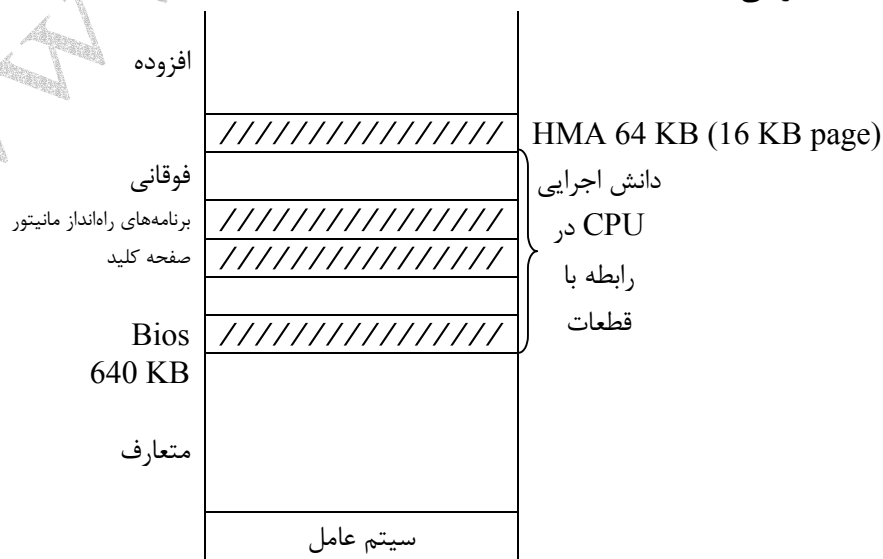
نکات مهم پیرامون حافظه RAM

- ۱- حافظه RAM یک برد الکترونیکی می‌باشد که با اجرا شدن برنامه سیستم عامل بخشهای گفته شده بالا بوجود می‌آید.
- ۲- بخشهای متعارف ، فوقانی و افزوده ، بخشهای منطقی و نرم‌افزاری می‌باشند و با وجود سیستم عامل ایجاد می‌گردند.
- ۳- این تقسیم‌بندی منطقی تنها در حافظه RAM کامپیوترهای IBM وجود دارد.
- ۴- در سیستم‌های IBM سه مُد آدرس‌دهی وجود دارد که عبارتند از : Real ، Protect و

Virtual

- ۵- اگر برنامه شما در حافظه متعارف قرار داشته باشد و محدوده آدرس‌دهی شما بین 0 و تا

- 640 کیلو بایت باشد آنرا آدرس دهی واقعی یا Real می گویند.
- ۶- اگر بخشی از برنامه شما در حافظه افزوده قرار داشته باشد و شما مجبور باشید برای آدرس دهی از فضای بالای 1 MB استفاده نمائید این حالت را آدرس دهی Protect می گویند.
- ۷- اگر حالت آدرس دهی شما جدا از دو مورد گفته شده و در فضاهای دیگری باشد آنرا مجازی یا Virtual می نامند.
- ۸- پس از بارگذاری سیستم عامل در حافظه RAM کنترل و ناظم سیستم ، سیستم عامل می باشد. به همین علت سیستم عامل را پل ارتباطی بین سخت افزار و نرم افزار می دانند.
- ۹- فایل Config.sys توسط کاربر ایجاد می شود و کاربر با استفاده از فرامین سیستم عامل پیکربندی های کامپیوتر را بر حسب نیاز خود انجام می دهد. در این فایل دستوراتی نظیر معرفی برنامه های راه انداز ، تعداد Buffer ، تعداد فایل های باز و همچنین فعال سازی حافظه های افزوده و فوقانی قرار دارد.
- ۱۰- دو تا از مهمترین دستوراتی که در فایل Config.sys وجود دارد دو دستور himem.sys و EMM386.exe می باشد. این دو دستور برای مدیریت فایل ها و حافظه RAM استفاده می گردد.
- ۱۱- هرگاه دو فرمان himem.sys و EMM386.exe را اجرا می کنیم شما توانسته اید از فضاهای خالی حافظه فوقانی و حافظه افزوده بهره ببرید. با این عمل فضاهای گفته شده مدیریتش در اختیار شما قرار می گیرد و CPU می تواند به دستورات درون این فضاها دسترسی داشته باشد.



۱۲- پس از Config.sys ، فایل Autoexec.bat اجرا می‌شود. این فایل نیز توسط کاربر ایجاد می‌شود و حاوی دستوراتی است که قرار است سیستم عامل در ابتدای روشن شدن کامپیوتر اجرا نماید.

۱۳- درون فایل Autoexec.bat فرامین اجرایی سیستم عامل قرار دارد. دستوراتی نظیر تعیین مسیر دستورات ، تعیین علامت اعلان Dos و غیره

تذکر: شما می‌توانید از بخشهای مختلف حافظه RAM استفاده نمائید. برای انجام این منظور شما باید بتوانید به آدرس فیزیکی و واقعی آن محل دسترسی داشته باشید.

۱۴- کلید برنامه‌هایی که در حافظه افزوده قرار دارند برای اجرا شدن به محیط شبیه‌سازی شده HMA می‌آیند در صورتیکه هسته اصلی برنامه‌ها در قسمت متعارف قرار می‌گیرند.

سؤال: برنامه راه‌انداز چیست؟

جواب: کلید برنامه‌هایی هستند که وظیفه دارند تبادل اطلاعات بین کامپیوتر و دستگاه مربوطه را فراهم نمایند. به عبارتی ساده‌تر برنامه‌های راه‌انداز از طرز کار و روش تبادل اطلاعات بین کامپیوتر و دستگاه مربوطه را درون خود دارند.

۱۵- هرگاه برنامه‌های راه‌انداز توسط دستور Device اجرا می‌گردند این برنامه‌ها در حافظه متعارف قرار می‌گیرند و چنانچه دستور Device high اجرا شود در حافظه فوقانی قرار می‌گیرد.

۱۶- در کامپیوترهای قدیمی‌تر ، حافظه‌های توسعه یافته نیز وجود داشته است. این حافظه‌ها بصورت اختیاری بوده‌اند که برنامه‌ها جهت اجرا شدن می‌توانستند از این فضا استفاده نمایند.

مثال مهم در مورد شناسایی سه بیت Ax و OF و CF

$$\begin{array}{r}
 +13 + \quad 00001101 + \\
 +9 \quad \quad 00001001 \\
 \hline
 +22 \quad \quad 00010110 \rightarrow +22
 \end{array}$$

$Ax = 1$
 $CF = 0$
 $OF = 0$

بیت علامت نشاندهنده عدد مثبت

$$\begin{array}{r}
 -13 + \quad 11110011 + \\
 +9 \quad \quad 00001001 \\
 \hline
 -4 \quad \quad 11111110 \rightarrow -4
 \end{array}$$

$Ax = 0$
 $CF = 0$
 $OF = 0$

بیت علامت نشاندهنده عدد منفی

$$\begin{array}{r}
 +13 + \quad 00001101 + \\
 -9 \quad \quad 11110111 \\
 \hline
 +4 \quad \quad 100000100 \rightarrow +4
 \end{array}$$

$Ax = 1$
 $CF = 1$
 $OF = 1$

بیت علامت نشاندهنده عدد مثبت
 CF = Invalid
 حالت CF را در نظر بگیرید

$$\begin{array}{r}
 -13 + \quad 11110011 + \\
 -9 \quad \quad 11110111 \\
 \hline
 -22 \quad \quad 11101010 \rightarrow -22
 \end{array}$$

$Ax = 0$
 $CF = 1$
 $OF = 1$

بیت علامت نشاندهنده عدد منفی
 CF = Valid
 حالت CF را در نظر بگیرید

تمرین : محاسبات فوق را بر روی عدد 23 و 13 اجرا کنید.

$$\begin{array}{r}
 +23 + \quad 00010111 + \\
 +13 \quad \quad 00001101 \\
 \hline
 +36 \quad \quad 00100100 \rightarrow +36
 \end{array}$$

$Ax = 1$
 $CF = 0$
 $OF = 0$

بیت علامت نشاندهنده عدد مثبت

$$\begin{array}{r}
 -23 + \quad 11101001 + \\
 +13 \quad \quad 00001101 \\
 \hline
 -10 \quad \quad 11110110 \rightarrow -10
 \end{array}$$

$Ax = 1$
 $CF = 0$
 $OF = 0$

بیت علامت نشاندهنده عدد منفی

$$\begin{array}{r}
 +23 + \quad 00010111 + \\
 -13 \quad \quad 11110011 \\
 \hline
 +10 \quad \quad 100001010 \rightarrow +10
 \end{array}$$

$Ax = 1$
 $CF = 1$
 $OF = 1$

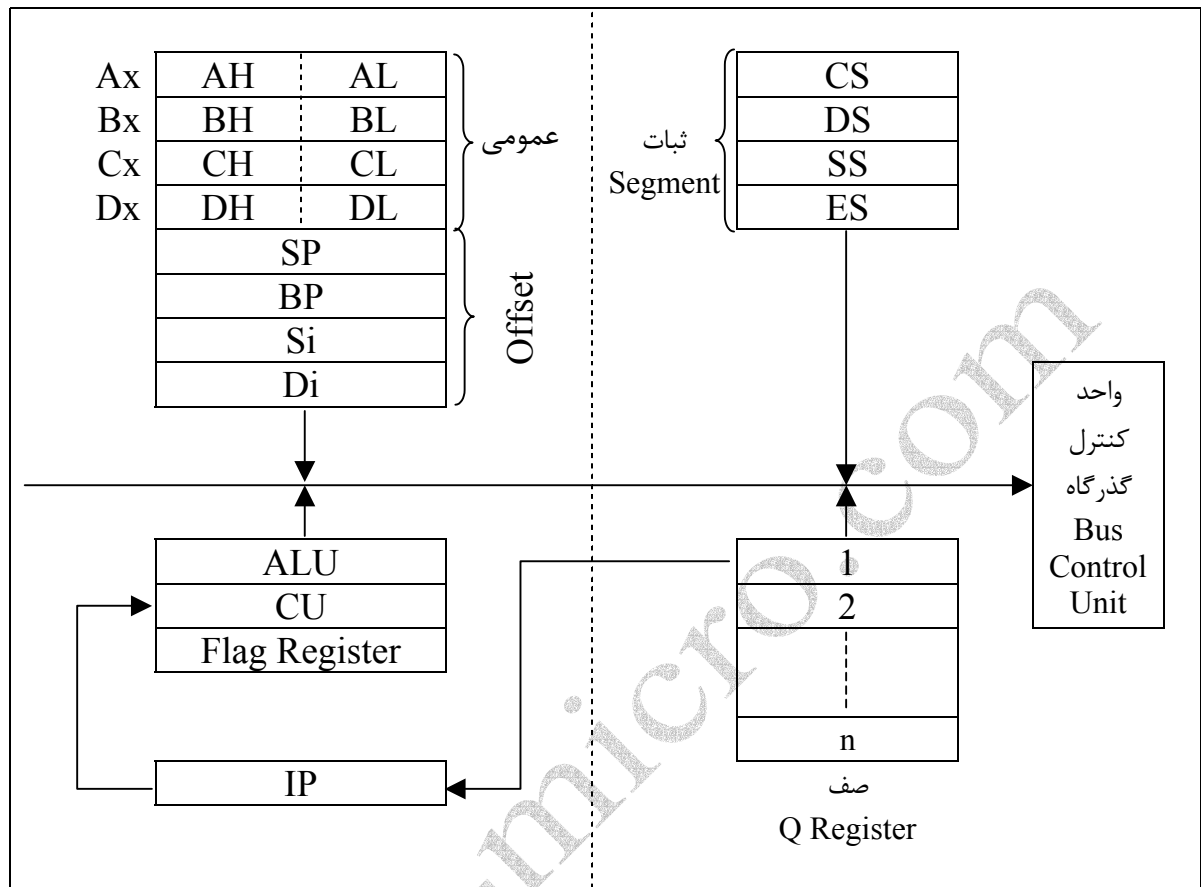
بیت علامت نشاندهنده عدد مثبت
 CF = Invalid
 حالت CF را در نظر بگیرید

$$\begin{array}{r}
 -23 + \quad 11101001 + \\
 -13 \quad \quad 11110011 \\
 \hline
 -36 \quad \quad 11011100 \rightarrow -36
 \end{array}$$

$Ax = 0$
 $CF = 1$
 $OF = 1$

بیت علامت نشاندهنده عدد منفی
 CF = Valid
 حالت CF را در نظر بگیرید

آشنایی جزئی و دقیق‌تر با CPU



واحد اجرایی EU

واحد گذرگاه (مسیر) ارتباطی BIU

CPU از لحاظ منطقی و از دید برنامه‌نویسی به دو منطقه تقسیم می‌شود. این تقسیم‌بندی بر اساس شرح وظایف قطعات داخل CPU می‌باشد. بدین منظور که ثبات‌های اجرایی و قطعات اجرایی که در محاسبات و اجرای دستورات دخالت دارند را بخش اجرایی می‌نامند و قطعاتی که در فعالیت اجرایی شرکت نمی‌کنند را قسمت ارتباطی می‌نامند.

تذکر: در شمای بالا قطعات و ثبات‌هایی را مشاهده می‌کنید که در زبان ماشین و زبان اسمبلی با آنها سر و کار داریم.

سؤال: از دید زبان ماشین، CPU دارای چه قسمت‌هایی است؟

الف) واحد اجرایی (EU) Execute Unit: این واحد حاوی قطعات و ثبات‌هایی می‌باشد که وظیفه دارد دستورات و فرامین را اجرا نماید. این بخش حاوی ثبات‌های عمومی، ثبات‌های افسست، ALU، CU، IP و ثبات Flag می‌باشد.

ب) واحد گذرگاه (مسیر) ارتباطی **Bus Interface Unit (BIU)**: این بخش حاوی ثبات‌ها و قطعاتی است که تنها ارتباط دهنده می‌باشند و همچنین تنها اطلاعات را درون خود نگهداری می‌کنند.

تذکر ۱: ثبات‌های دیگری نیز در مدل‌های جدیدتر CPU وجود دارد که هر کدام فعالیت خاصی را بر عهده دارند.

تذکر ۲: **ALU** تنها می‌تواند محاسبات عددی، منطقی و شیفت دادن‌ها را اجرا نماید.

سؤال: ثبات چیست؟

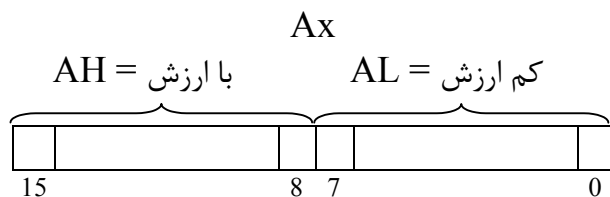
جواب: حافظه‌ای است موقتی، سریع و گران قیمت که اندازه آن می‌تواند ۲ بایت یا ۴ بایت باشد و در مدل‌های قدیمی‌تر ۱ بایت بوده است.

تذکر ۱: ثبات‌ها هر کدام دارای اسمی منحصر به فرد می‌باشند و توسط این اسم شناسایی می‌شوند.

تذکر ۲: با قطع برق، اطلاعات درون ثبات‌ها از بین می‌رود.

تذکر ۳: داده درون ثبات می‌تواند کاراکتر و یا عدد باشد. برداشت از آن به دیدگاه شما بستگی دارد.

تذکر ۴: تمام ثبات‌های عمومی از دو بخش تشکیل شده‌اند.



۱- بخش کم ارزش

۲- بخش با ارزش

تذکر ۱: ثبات‌ها دارای دو کاربرد می‌باشند: کاربردهای عمومی و کاربردهای اختصاصی. برای مثال شما می‌توانید از **Ax** جهت قرار داد مقادیر و انجام محاسبات استفاده نمایید. در صورتیکه سیستم نیز از **Ax** استفاده‌هایی را می‌کند.

تذکر ۲: ثبات‌ها چهار نوع می‌باشند.

۱- ثابت‌های عمومی: این ثابت‌ها شامل **Ax**, **Bx**, **Cx** و **Dx** می‌باشند.

۲- ثابت‌های افسست: ثابت‌هایی نظیر **Sp**, **Bp**, **Si**, **Di** و **IP** می‌باشند.

۳- ثابت **Flag**: این ثابت وضعیت عملکرد CPU را به شما نشان می‌دهد.

۴- ثابت **Segment**: این ثابت‌ها عبارتند از **CS**, **DS**, **SS** و **ES**. این ثابت‌ها حاوی آدرس شروع **Segment** می‌باشند.

Segment چیست؟ بخشی از حافظه متعارف می‌باشد که یک فضای محدود و مرزبندی شده در دل یک پاراگراف می‌باشد. اندازه **Segment**‌ها 64 کیلو بایت می‌باشد که آدرس شروع آنها عددی

است که به عدد 16 (Hex 10) بخش پذیر است.

تذکر ۱: شروع هر Segment با مرز یک پاراگراف یکی می باشد.

تذکر ۲: درون هر ثبات Segment عددی وجود دارد که حاصل جمع این عدد با مقدار افسست (برای

مثال IP) به CPU محل واقعی و فیزیکی را نشان می دهد.

تذکر ۳: در حافظه ، داده های با ارزش در آدرس های پایین قرار دارند و داده های کم ارزش در

آدرس های بالایی قرار دارند.

اسمبلی	mov	Ax	,	کم با ارزش ارزش 1304
ماشین	B8	04		13

	B8	EF05	
کم ارزش	04	EF04	بالا
با ارزش	13	EF03	پایین

آدرس

تذکر ۱: سگمنت ها فضاهای کاری می باشند. بدین معنی که درون این فضاها کلیه دستورات ، فرامین

و داده ها قرار داده می شوند.

تذکر ۲: برای برنامه نویسی به زبان ماشین باید ابتدا سگمنت هایی را از سیستم عامل تقاضا نمائید و

سپس سیستم عامل از درون حافظه RAM (در بخش متعارف) فضایی را به ما اختصاص می دهد (که

کجا بودن آن زیاد مهم نیست) و به اندازه 64 کیلو بایت و در مبنای 16 می باشد.

تذکر ۳: هر سگمنت دارای یک آدرس شروع می باشد.

انواع سگمنت

۱. Code Segment: ناحیه ای است که درون آن کلیه دستورات اجرایی برنامه و همچنین

کلیه فرامین در آن نوشته می‌شود (همان نقش **Begin** و **End** را در پاسکال بازی می‌کند).

۲. **Data Segment**: سگمنتی است که در آن کلیه ثابت‌ها، متغیرها و داده‌ها تعریف می‌گردد.

تذکر: امکان دارد در برخی از برنامه‌ها **Data Segment** نداشته باشیم اما حداقل یک **Code Segment** داریم.

۳. **Stack Segment**: سگمنتی است که در آن کلیه آدرس‌های رفت و بازگشت زیربرنامه‌ها ذخیره می‌گردد و همچنین از آن برای انتقال اطلاعات به درون زیربرنامه نیز استفاده می‌گردد.

تذکر: از **Stack Segment** می‌توانیم جهت جابجایی داده‌ها و اطلاعات استفاده نمائیم.

۴. **Extra Segment**: این سگمنت اضافی برای انجام محاسبات بر روی رشته‌ها و آرایه‌ها استفاده می‌شود.

تذکر ۱: همانطور که می‌دانید سگمنت‌ها دارای آدرس شروع می‌باشند و بطور تصادفی در یکی از قسمت‌های حافظه متعارف قرار دارند.

تذکر ۲: شما می‌توانید در یک برنامه، یک یا چند سگمنت تقاضا کنید اما همیشه حداقل یک **Code Segment** دارید.

تذکر ۳: شما می‌توانید در یک برنامه یک یا چند **Code Segment** داشته باشید و همینطور سگمنت‌های دیگر اما **Stack Segment** یکی است (فرض کنید این سگمنت به **Stack Segment** سیستم متصل باشد).

تذکر ۴: برای اینکه اطلاعات را درون **Stack Segment** قرار دهیم از دستور **Push** استفاده می‌کنیم و همچنین برای اینکه اطلاعات را از درون **Stack** به بیرون انتقال دهیم از دستور **pop** استفاده می‌کنیم.

تذکر ۵: آدرس شروع سگمنت‌ها ۴ رقمی است و در مبنای ۱۶ می‌باشد.

سؤال: افس (offset) چیست؟

جواب: به معنی تفاوت مکانی و یا انطباق می‌باشد. ثبات‌های افس تعیین مکان می‌کنند و همانند یک شمارنده حاوی آدرسی می‌باشند.

تذکر ۱: آدرس سگمنت‌ها مطلق (و ثابت) می‌باشند. در صورتیکه هر سطر دستور درون سگمنت‌ها دارای یک آدرس می‌باشد (نسبی نسبت به سگمنت). این آدرس عددی است ۴ رقمی در مبنای ۱۶ که بین ۰۰۰۰ و ffff می‌باشد و همین رنج در مبنای ۱۰ بین ۰ تا ۶۵۵۳۵ می‌باشد.

تذکر ۲: پس از تقاضای سگمنت ما باید با دستور Assume آدرس شروع هر سگمنت را به درون ثبات سگمنت مربوطه‌اش بریزیم.

تذکر ۳: باید آدرس شروع Code Segment را درون ثبات سگمنت CS بریزیم.

تذکر ۴: باید آدرس شروع Data Segment را درون ثبات سگمنت DS بریزیم.

تذکر ۵: باید آدرس شروع Stack Segment را درون ثبات سگمنت SS بریزیم.

تذکر ۶: باید آدرس شروع Extra Segment را درون ثبات سگمنت ES بریزیم.

تذکر ۷: IP ثابتی است که جزء ثبات‌های افست می‌باشد و همانند یک شمارنده، آدرس نسبی هر دستور را نشان می‌دهد.

تذکر ۸: برای بدست آوردن آدرس فیزیکی و واقعی هر دستور در حافظه RAM شما باید الگوی زیر را اجرا نمایید.

$$0 + \text{آدرس شروع هر سگمنت} \quad n n n n 0 + \quad 0 1 0 0 0 +$$

$$\text{مقدار ثبات IP (نسبی)} \quad m m m m \quad 0 0 0 1$$

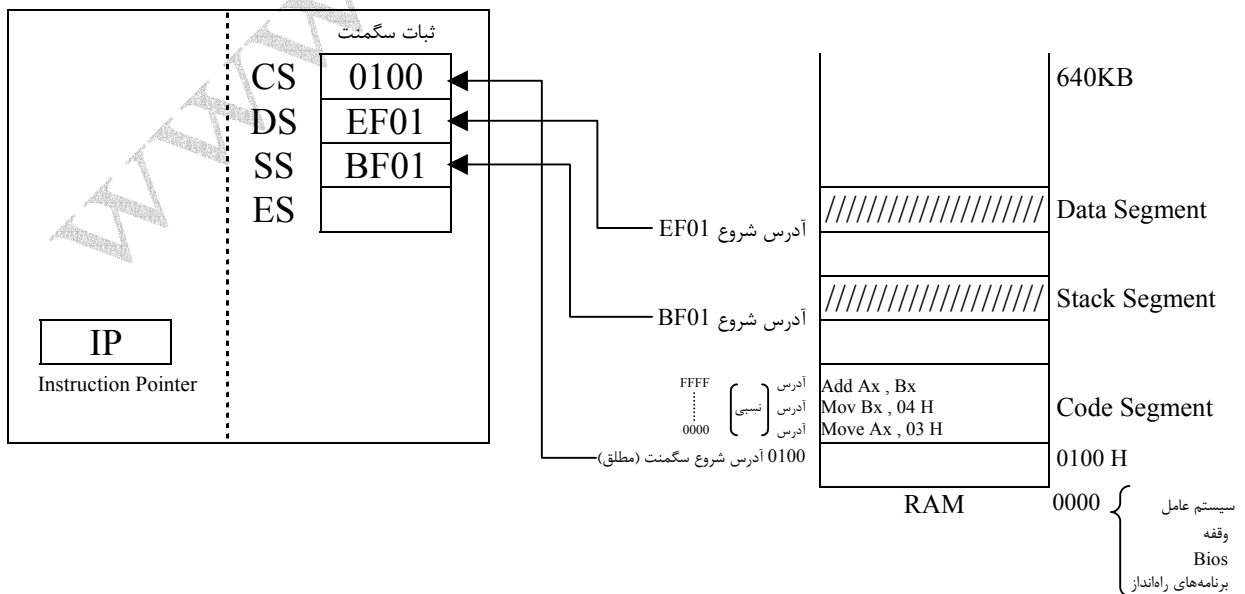
آدرس فیزیکی هر دستور

آدرس فیزیکی

آدرس فیزیکی اولین دستور

تذکر ۱: هر سگمنت دارای یک افست مخصوص بخود است و افست مربوط به Code Segment نیز افست IP می‌باشد.

تذکر ۲: علت اضافه کردن یک صفر به جلوی آدرس سگمنت به خاطر بوجود آمدن فضای یک مگابایتی حافظه RAM می‌باشد. چون با قرار دادن یک صفر، تعداد ارقام در مبنای 2 بیست رقم خواهد شد.



تذکر ۱: هر بایت حافظه (درون سگمنت) دارای یک آدرس چهار رقمی می‌باشد. به عبارتی بایت‌های حافظه قابل آدرس‌دهی می‌باشد.

تذکر ۲: کلیه دستورات زبان اسمبلی به کد ماشین ترجمه می‌شود و کدهای ماشین درون حافظه قرار می‌گیرند.

تذکر ۳: هر کد ماشین از دو رقم در مبنای 16 تشکیل شده است.

تذکر ۴: مقدار IP از صفر شروع می‌شود اما دو تا دو تا افزایش می‌یابد زیرا خواندن اطلاعات از درون حافظه بصورت دو بایتی انجام می‌شود (در سیستم‌های قدیمی تر دو تا یک بایتی و در سیستم‌های جدیدتر سه بایتی نیز می‌باشد).

0005			
0004		اسمبلی	mov Ax , 03 H
0003	8B	کد ماشین	(B8 03) ₁₆
0002	04	Segment	داده OP - Code
0001	B8	سیستم عامل	اسمبلی
0000	03		mov Bx , 04 H
0100 H		کد ماشین	8B 04

یک بایت داده

تذکر ۱: مقدار IP دو تا دو تا افزایش می‌یابد.

تذکر ۲: اجرای دستورات در سیستم‌های قدیمی تر در دو Fetch انجام می‌گردد. Fetch اول گرفتن دستور یا Op - Code و Fetch دوم دریافت داده‌ها و یا اطلاعات.

تذکر ۳: Fetch به معنی گرفتن اطلاعات از حافظه RAM و انتقال آن به درون ثبات‌های CPU است.

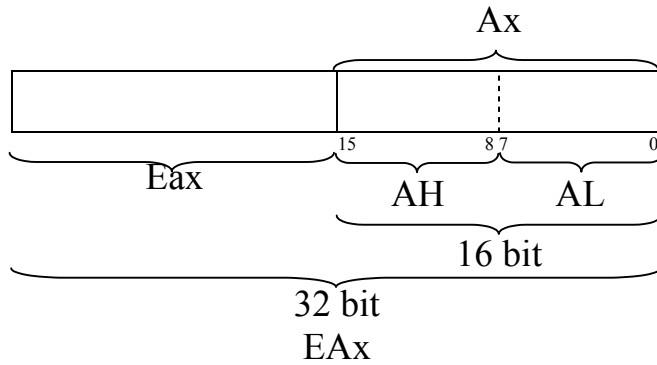
آشنایی با ثبات‌های درون CPU

۱- ثبات Ax: این ثبات به مفهوم آکومولاتور (Accumulator) می‌باشد که در محاسبات

شرکت داده می‌شود و همچنین در آن محتوای نتایج محاسبات نیز ذخیره می‌گردد و از آن

برای انتقال اطلاعات و داده‌ها به درون وقفه‌ها (Interrupt) استفاده می‌گردد.

تذکر: این ثبات 16 بیتی می‌باشد و در سیستم‌های جدید 32 بیتی است.



۲- **ثبات Base Register (Bx)**: این ثبات به عنوان ثبات پایه در آدرس‌دهی‌ها استفاده می‌گردد و همچنین برای انجام محاسبات از آن استفاده می‌گردد.

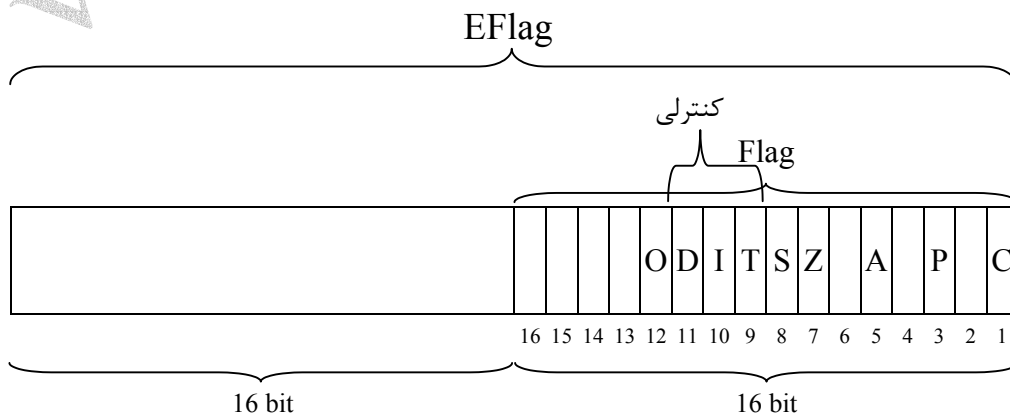
۳- **ثبات Counter Register (Cx) (ثبات شمارنده)**: این ثبات جهت انجام محاسبات استفاده می‌گردد و همچنین در ایجاد حلقه‌ها بعنوان متغیر شمارنده از آن استفاده می‌گردد.

۴- **ثبات Destination Register (Dx)**: این ثبات کاربردهای متنوع و گوناگونی دارد:

- برای انجام محاسبات
- بیت‌های اضافی که در AX ذخیره نمی‌شوند در این ثبات قرار می‌گیرند.
- برای انجام محاسبات در رشته‌ها و آرایه‌ها

تذکر: تمام ثبات‌های عمومی گفته شده دارای ساختاری همانند AX در کامپیوترهای جدید دارند.

۵. **ثبات Flag**: وضعیت پردازش را به شما نشان می‌دهد بدین معنی که CPU پس از پردازش وضعیت اجرای دستورات را در این ثبات به شما نشان می‌دهد. این ثبات در سیستم‌های جدید 32 بیتی می‌باشد. در صورتیکه در سیستم‌های قدیمی‌تر 16 بیت بوده است. 9 بیت آن فعال که 6 بیت آن مربوط به وضعیت پردازش دستورات است و 3 بیت آن جزء بیت‌های کنترلی است.



✓ **بیت اول** ← **Cary Flag (CF)**: این بیت نشان دهنده رقم انتقالی در انجام محاسبات می‌باشد. رقم انتقالی بی‌تی است که از بیت هفتم به هشتم منتقل شده است یا اینکه از بیت پانزدهم به شانزدهم منتقل شده است (آخرین بیت نقلی می‌باشد). این بیت همچنین نشاندهنده مقدار شیفت می‌باشد که دو وضعیت دارد. اگر رقم نقلی نباشد صفر است و اگر انتقال انجام شده باشد یک است که کد آنها بصورت زیر است

0 → NC

1 → PC

✓ **بیت سوم** ← **Parity Flag (PF)**: این بیت نشاندهنده بیت توازن می‌باشد. یعنی اینکه اطلاعات درست منتقل شده‌اند یا خیر. این بیت همچنین نشاندهنده تعداد بیت‌های شیفت داده شده می‌باشد. اگر تعداد بیت‌های شیفت داده شده زوج باشد مقدار آن یک است و چنانچه تعداد شیفت‌ها فرد باشد مقدار آن صفر است.

0 → PO (فرد)

1 → PE (زوج)

✓ **بیت پنجم** ← **Auxiliary Flag (AF)**: این بیت نشاندهنده رقم نقلی از بیت سوم به بیت چهارم می‌باشد. اگر رقم نقلی نداشته باشید و یا اینکه تعداد شیفت‌ها زوج باشد مقدار آن صفر است در غیر اینصورت یک است.

0 → NA (زوج)

1 → PA (فرد)

✓ **بیت هفتم** ← **Zero Flag (ZF)**: این بیت به شما نشان می‌دهد که نتیجه محاسبات یا نتیجه مقایسه برابر صفر شده است یا خیر. اگر نتیجه محاسبات صفر باشد و یا اینکه دو داده با هم مساوی باشند مقدار این بیت یک است و در غیر اینصورت صفر است. مثال :

CMP Ax , Bx

ZF = 1 اگر محتوای Bx و Ax برابر باشد.

محاسبه $\left\{ \begin{array}{l} AL = 3 \rightarrow 0000\ 0011 \\ BL = 4 \rightarrow 0000\ 0100 \\ \hline 0000\ 0111 \end{array} \right\} \Rightarrow ZF = 0$ مقدار نتیجه برابر صفر نیست

مقایسه $\left\{ \begin{array}{l} AL = 5 \rightarrow 0000\ 0101 \\ BL = 6 \rightarrow 0000\ 0110 \end{array} \right\} \Rightarrow ZF = 0$ دو داده برابر نیستند

✓ بیت هشتم ← **Sign Flag (SF)**: این بیت نشان دهنده همان بیت علامت در نتیجه محاسبات می‌باشد. این بیت دو وضعیت دارد. اگر صفر باشد نشان دهنده مثبت بودن نتیجه محاسبات می‌باشد و چنانچه یک باشد نشان دهنده منفی بودن محاسبات می‌باشد.

✓ بیت نهم ← **Trace / Trap Flag (TF)**: با استفاده از این بیت ما می‌توانیم مشخص نمائیم اجرای دستورات بصورت تک مرحله‌ای انجام شود و یا اینکه بصورت یکی یکی و خط به خط اجرا شود. اگر مقدار این بیت یک باشد اجرای دستورات یکی یکی می‌باشد و چنانچه صفر باشد دستورات یکجا اجرا می‌شوند.

✓ بیت دهم ← **Interrupt Flag (IP)**: این بیت برای شما مشخص می‌کند که سیستم می‌تواند به وقفه‌ها پاسخ دهد یا خیر. اگر مقدار این بیت صفر باشد سیستم نمی‌تواند به وقفه‌ها پاسخ دهد و چنانچه یک باشد سیستم می‌تواند به وقفه‌ها پاسخ دهد.

✓ بیت یازدهم ← **Direction Flag (DF)**: با استفاده از این بیت، جهت انجام عملیات‌ها بر روی رشته‌ها و آرایه‌ها را مشخص می‌کنیم. بدین معنی که با استفاده از این بیت اولین خانه آرایه را ما می‌توانیم مشخص نمائیم. اگر این بیت مقدار یک باشد جهت انجام عملیات‌ها (مقایسه یا شیفت) از سمت راست به چپ و چنانچه صفر باشد انجام عملیات‌ها از سمت چپ به راست می‌باشد.

✓ بیت دوازدهم ← **Over Flow Flag (OF)**: این بیت نشان‌دهنده مقدار سرریزی بیت‌ها می‌باشد که همان حالت Over Flow است. اگر مقدار سرریزی داشته باشید مقدار این بیت یک است و اگر نداشته باشید مقدار این بیت صفر است.

0 → NO

1 → PO

تمرین: با انجام محاسبه زیر اثر بیت‌ها در ثبات Flag را اعلام نمائید.

$$66 - \quad 01000010 + \quad CF = 1$$

$$22 \quad 11101010 \quad Ax = 0$$

$$44 \quad 100101100 \quad OF = 1$$

$$SF = 0$$

$$ZF = 0$$

تمرین : نتیجه محاسبه زیر را در ثبات Flag نشان دهید و نتیجه را به 32 مقایسه کنید.

$$\begin{array}{r}
 42 - \quad 00101010+ \\
 28 \quad \quad 11100100 \\
 \hline
 14 \quad \quad 100001110
 \end{array}$$

				O	D	I	T	S	Z		A		P		C
				1	C	C	C	0	0		0		0		1

16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 غیر معتبر

مقایسه با 32

$$\left. \begin{array}{l}
 \text{نتیجه} \quad 100001110 \\
 32 \quad \quad 00100000
 \end{array} \right\} \text{برابر نیستند}$$

				O				S	Z		A		P		C
				0	D	I	T	0	0		0		0		0

16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

تمرین : بیت‌های Flag را پس از انجام محاسبه زیر نشان دهید.

$$\text{Mov AL, 4} \quad 4 = 00000100$$

				O				S	Z		A		P		C
				0	D	I	T	0	0		0		0		0

16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

۶. ثبات **Start Index (SI)** : این ثبات جزء ثبات‌های اندیس می‌باشد که دارای 16 بیت

است و مدل جدیدتر آن ESI و دارای 32 بیت می‌باشد. از این ثبات برای آدرس‌دهی‌ها و همچنین اندیس آرایه‌ها استفاده می‌گردد. کاربرد خاص آن مبدأی است برای انجام

عملیات بر روی رشته‌ها

تذکر خیلی مهم : ثبات SI با ثبات سگمنت DS ترکیب می‌شود.

۷. ثبات **Destination Index (DI)**: این ثبات به عنوان ثبات اندیس می‌باشد که دارای 16 بیت است و در کامپیوترهای جدید بصورت EDI و 32 بیتی می‌باشد. از این ثبات جهت انجام عملیات بر روی آرایه‌ها و رشته‌ها استفاده می‌گردد. کاربرد خاص آن بعنوان مقصد جهت انجام عملیات بر روی رشته‌ها می‌باشد.

تذکر: این ثبات نیز با ثبات سگمنت DS ترکیب می‌گردد.

سؤال: اجرای دستورات در زبان ماشین در چند مرحله اجرا می‌گردد؟ بدین مفهوم که اجرای دستورات در هر Fetch چگونه است؟

جواب: اجرای دستورات در سه مرحله اجرا می‌گردد:

- **مرحله اول:** دریافت Op - Code دستورات و یا داده‌ها و قرار دادن آن در صف اجرای دستورات
- **مرحله دوم:** رمزگشایی دستور و شناسایی دستور می‌باشد. این عمل توسط CU صورت می‌پذیرد.
- **مرحله سوم:** اجرای دستورات که توسط ALU انجام می‌گیرد.

سؤال: حافظه‌ها دارای چند خاصیت مشترک می‌باشند؟

جواب: تمام حافظه‌ها باید خاصیت‌های زیر را دارا باشند:

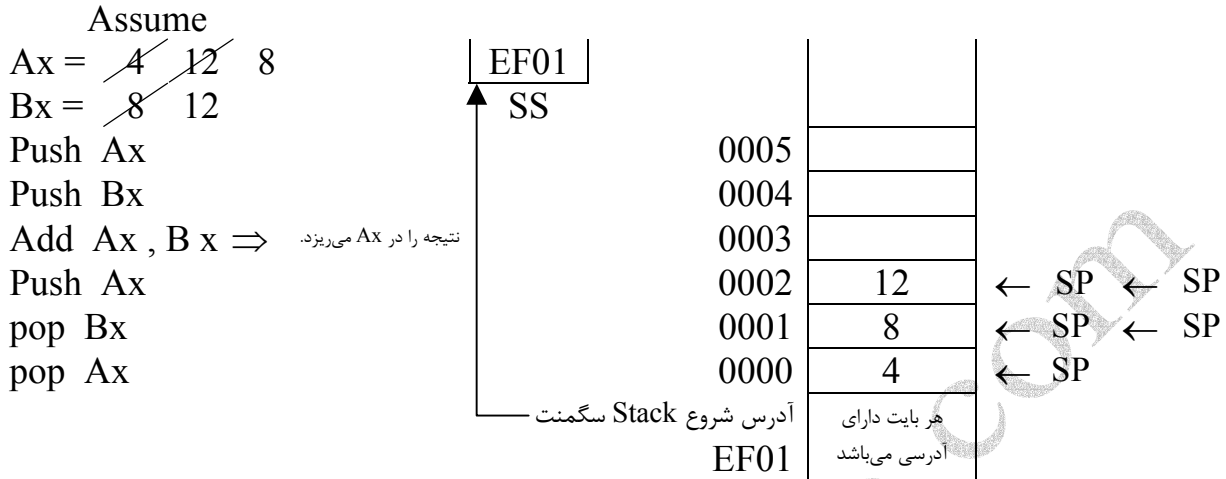
- ۱- قابل خواندن و نوشتن اطلاعات باشند.
- ۲- تمام حافظه‌ها باید دارای ظرفیت یا گنجایش باشند.
- ۳- تمام بایت‌های حافظه باید قابل آدرس‌پذیری و دستیابی باشند که در برخی از حافظه‌ها واحدهای بزرگتری قابل آدرس‌دهی می‌باشند.
- ۴- زمان دسترسی باید در حافظه‌ها وجود داشته باشد.
- ۵- در تمام حافظه‌ها باید سرعت انتقال وجود داشته باشد.
- ۶- همه حافظه‌ها باید بتوانند اطلاعات و داده‌ها را بطور پایدار در خود نگهداری نمایند.

آشنایی جزئی‌تر با Stack

Stack همانند یک حافظه عمل می‌کند. بطوریکه در آن می‌توانیم اطلاعات را به طور موقت ذخیره نمائیم. این سگمنت دارای یک آدرس شروع می‌باشد که برای بدست آوردن آدرس فیزیکی با مقدار SP ترکیب می‌گردد.

تذکر ۱: تنها دستوری که اطلاعات و داده‌ها را به درون Stack می‌ریزد دستور Push می‌باشد و تنها دستوری که از Stack خارج می‌کند دستور pop می‌باشد.

تذکر ۲: ثابت SP نشان دهنده محلی است که در آن اطلاعات می‌ریزیم و یا اطلاعات بر می‌داریم.



ثبات سگمنت +	SS +	EF010 +	EF010 +
Offset	SP	0000	0001
آدرس فیزیکی	(آدرس واقعی) EF010	EF010	EF011

تذکر: دو ثابت سگمنت دیگر وجود دارد بنامهای GS و FS که این ثابت‌ها در موارد مورد نیاز استفاده می‌شوند و می‌توانید از آنها جهت ثابت‌های سگمنت Code و یا ثابت‌های اضافه استفاده نمائید.

۸. ثابت Stack Pointer (SP) (ثبات اشاره‌گر پشته): این ثابت 16 بیتی می‌باشد که

برای افسست در Stack Segment استفاده می‌شود. این ثابت در کامپیوترهای جدید بصورت

32 بیتی می‌باشد که حرف اول آن E قرار می‌گیرد (ESP).

تذکر: این ثابت جزو ثابت‌های اشاره‌گر می‌باشد.

۹. ثابت Base Pointer (BP): این ثابت 16 بیتی می‌باشد که برای انتقال اطلاعات به

درون Stack استفاده می‌شود. داده‌هایی که به درون Stack منتقل می‌شوند می‌توانند داده

و یا آدرس باشند.

تذکر ۱: این ثابت به عنوان افسست یا ثابت SS ترکیب می‌گردد.

تذکر ۲: ثابت BP به عنوان ثابت پایه و اصلی می‌تواند با DI و SI ترکیب شود.

آشنایی با Interrupt ها (وقفه‌ها)

وقفه چیست؟

برنامه‌ای است که از چند دستورالعمل تشکیل شده است که یک هدفی را دنبال می‌کند. در واقع وقفه‌ها برنامه‌هایی حاضر و آماده می‌باشند که باید یک عمل خاصی را اجرا نمایند.

چند نوع وقفه وجود دارد؟

دو نوع وقفه وجود دارد :

۱- وقفه‌های سخت افزاری

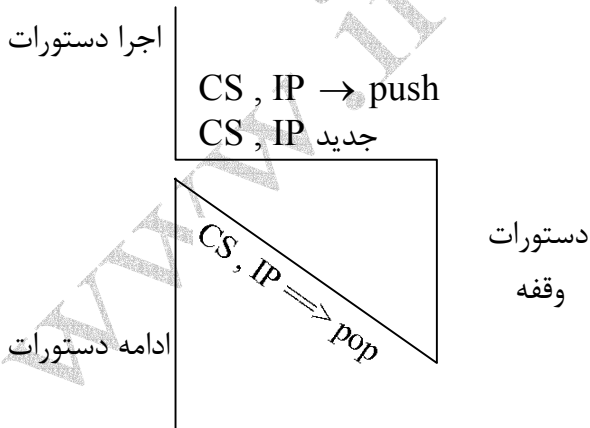
۲- وقفه‌های نرم افزاری

سؤال : وقفه‌های سخت افزاری چیست؟

جواب : برنامه‌هایی هستند که توسط CPU اجرا می‌گردند و اجرا شدن آنها منوط به دریافت سیگنالی توسط CPU می‌باشد.

تذکر : پالس‌هایی که برای انجام وقفه‌ها می‌باشند از مسیر IRQ به CPU منتقل می‌شوند.

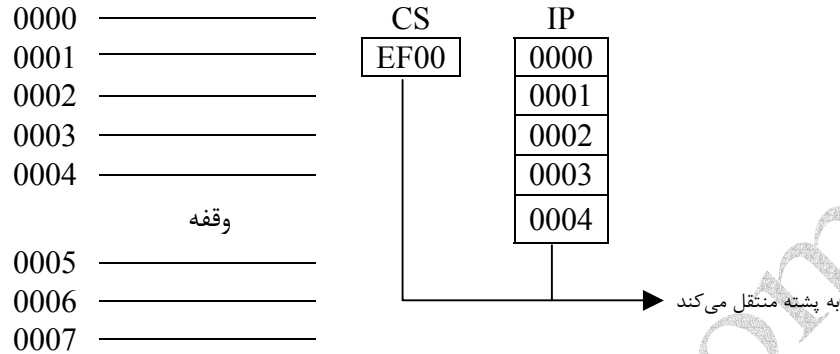
تذکر مهم : هرگاه وقفه‌ای را از CPU تقاضا می‌کنیم ، CPU اجرای برنامه‌های خود را متوقف کرده و دستورات درون وقفه را تا رسیدن به دستور IRET اجرا می‌کند و سپس به درون دستورات در حال اجرا باز می‌گردد. همانند الگوی زیر :



تذکر : هرگاه CPU و مجری (سیستم عامل) برنامه‌های زبان ماشین به دستور وقفه می‌رسند مقدار CS و IP را در Stack ذخیره نموده و محل وقفه را در حافظه RAM شناسایی می‌کند. چونکه متن برنامه‌های وقفه در یک سگمنت دیگری قرار دارد سیستم مترجم این آدرس‌های جدید را به ثبات‌های CS و IP منتقل می‌کند تا سیستم در مکان‌یابی اجرای دستورات دچار مشکل نشود. پس از اجرای دستورات درون وقفه و با رسیدن به دستور IRET دوباره سیستم مترجم مقادیر CS و IP که در

Stack ذخیره شده است را برداشته و به درون دو ثبات CS و IP می‌ریزد تا اجرای دستورات برنامه اصلی ادامه یابد.

شروع سگمنت EF00

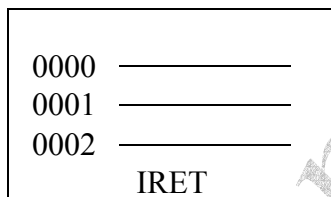


آدرس شروع پشته
BF00

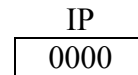


SS SP
 با دیدن دستور وقفه

آدرس شروع CF00



Code Segment



پایان

هرگاه به دستور IRET رسید باید ابتدا محتوای درون Stack را به CS و IP منتقل کند تا بتواند ادامه برنامه را اجرا نماید.
 دستور فراخوانی وقفه‌ها بصورت زیر می‌باشد.

Int H شماره وقفه

تذکر ۱: بر خلاف هِدِر فایل‌ها که دارای اسم هستند وقفه‌ها شماره‌بندی شده هستند.
 تذکر ۲: به زیربرنامه و امکاناتی که درون هر وقفه وجود دارد سرویس می‌گویند و سرویس‌ها نیز شماره‌بندی شده هستند.

تذکر ۳: هرگاه به وقفه‌ای نیاز داشته باشیم (در بین دستورات) باید با استفاده از دستور int آنرا فراخوانی نمائیم.

تذکر ۴: شماره سرویس را باید با دستور Mov درون ثبات AH قرار دهیم.

تذکر ۵: ساختار و عملکرد وقفه‌ها همانند توابع کتابخانه‌ای می‌باشند. یعنی اینک برنامه‌نویس باید

شماره سرویس ، اطلاعات ورودی به درون سرویس و عمل فراخوانی را انجام دهد و همه وقفه‌ها اطلاعات خروجی خود را درون ثبات‌ها می‌ریزد.

```

{
  int شماره وقفه H
  {
    Mov AH , 01 H    مشخص کردن شماره سرویس
      محل تعیین اطلاعات ورود به سرویس
  }
  int 21 H
  اطلاعات خروجی را درون ثبات‌ها می‌ریزد.

```

(مثال)

```

Scanf {
  Mov AH , 01 H
  یک کاراکتر از صفحه کلید دریافت می‌کند و کد اسکی آنرا درون ثبات AL می‌ریزد
  int 21 H

```

```

← شماره سرویس      Mov AH , 02 H    برای نمایش دادن کاراکتر و داده‌ها روی صفحه مانیتور
← اطلاعات ورودی به سرویس      Mov DL , 3
← فراخوانی وقفه      int 21 H

```

پروژه اول : برنامه‌ای بنویسید که ۱۳ عدد را از ورودی بخواند (یک رقمی) و میانگین آنرا چاپ کند.

پروژه دوم : مربع گرافیکی ترسیم شده را با زدن کلید Enter حرکت دهید و با زدن کلید Space متوقف کنید.

پروژه سوم : هفت کاراکتر را از ورودی بخوانید و آنرا مرتب چاپ کنید.